

BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBBBBBBBBBBB		AAAAAAA		SSSSSSSSSS		RRRRRRRRRR		TTTTTTTTTTTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA	SSS		RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRRRRRRRRR		TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAAAAAAAAAAA			SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBB	BBB	AAA	AAA		SSS	RRR	RRR	TTT		LLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL
BBBBBBBBBBBB		AAA	AAA	SSSSSSSS		RRR	RRR	TTT		LLLLLLLLLLLL


```
BBBBBBBBB      AAAAAA      SSSSSSSS      EEEEEEEEEE      DDDDDDDD      IIIIII      TTTTTTTTTT
BBBBBBBBB      AAAAAA      SSSSSSSS      EEEEEEEEEE      DDDDDDDD      IIIIII      TTTTTTTTTT
BB          BB  AA          AA  SS          EE          DD          DD  II          TT
BB          BB  AA          AA  SS          EE          DD          DD  II          TT
BB          BB  AA          AA  SS          EE          DD          DD  II          TT
BB          BB  AA          AA  SS          EE          DD          DD  II          TT
BBBBBBBBB      AA          AA  SSSSSS      EEEEEEEE      DD          DD  II          TT
BBBBBBBBB      AA          AA  SSSSSS      EEEEEEEE      DD          DD  II          TT
BB          BB  AAAAAAAAAA      SS          EE          DD          DD  II          TT
BB          BB  AAAAAAAAAA      SS          EE          DD          DD  II          TT
BB          BB  AA          AA  SSSSSS      EEEEEEEE      DD          DD  II          TT
BB          BB  AA          AA  SSSSSS      EEEEEEEE      DD          DD  II          TT
BBBBBBBBB      AA          AA  SSSSSSSS      EEEEEEEEEE      DDDDDDDD      IIIIII      TTTTTTTTTT
BBBBBBBBB      AA          AA  SSSSSSSS      EEEEEEEEEE      DDDDDDDD      IIIIII      TTTTTTTTTT

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL          II          SS
LL          II          SS
LL          II          SS
LL          II          SS
LL          II          SSSSSS
LL          II          SSSSSS
LL          II          SS
LL          II          SS
LL          II          SS
LL          II          SS
LLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLL      IIIIII      SSSSSSSS
```

BAS\$EDIT
Table of contents

(2) 54
(3) 123

DECLARATIONS
BAS\$EDIT

- Perform BASIC EDIT\$ function


```
0000 1      .TITLE BAS$EDIT
0000 2      .IDENT /1-009/
0000 3
0000 4      ; File: BAS$EDIT.MAR Edit: SBL1009
0000 5      ;*****
0000 6      ;
0000 7      ; COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      ; ALL RIGHTS RESERVED.
0000 10     ;
0000 11     ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12     ; ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13     ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14     ; COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15     ; OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16     ; TRANSFERRED.
0000 17     ;
0000 18     ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19     ; AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20     ; CORPORATION.
0000 21     ;
0000 22     ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23     ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24     ;
0000 25     ;*****
0000 26     ;
0000 27     ;
0000 28     ;
0000 29     ;++
0000 30     ; FACILITY: BASIC code support
0000 31     ;
0000 32     ; ABSTRACT:
0000 33     ;
0000 34     ;     This module implements the BASIC-PLUS-2 EDIT$ function.
0000 35     ;
0000 36     ; ENVIRONMENT: User Mode, AST Reentrant
0000 37     ;
0000 38     ;--
0000 39     ; AUTHOR: R. WILL, CREATION DATE: 2-May-79
0000 40     ;
0000 41     ; MODIFIED BY:
0000 42     ;
0000 43     ; R. Will, : VERSION 1
0000 44     ; 1-001 - Original
0000 45     ; 1-002 - Fix order of source and dest str$ in TRM. RW 16-May-79
0000 46     ; 1-003 - Enable IV in entry masks. RW 23-May-79
0000 47     ; 1-004 - Change calls to STR$COPY. JBS 16-JUL-1979
0000 48     ; 1-005 - Make PIC. RW 13-Sept-79
0000 49     ; 1-006 - String cleanup, call STR$TRM. RW 31-Oct-79
0000 50     ; 1-007 - Remove BASTRM. RW 13-NOV-79
0000 51     ; 1-008 - Add support for international character set. ML 21-Jun-82
0000 52     ; 1-009 - Use general addressing mode for all externals. SBL 3-Nov-1982
```


DECLARATIONS

```
0000 54 .SBTTL DECLARATIONS
0000 55 :
0000 56 : INCLUDE FILES:
0000 57 :
0000 58
0000 59 $DSCDEF ; Define descriptor offsets
0000 60
0000 61 :
0000 62 : EXTERNAL DECLARATIONS:
0000 63 :
0000 64 .DSABL GBL ; Prevent undeclared
0000 65 ; symbols from being
0000 66 ; automatically global.
0000 67
0000 68 .EXTRN STR$COPY_DX_R8 ; copy input string to temp
0000 69 ; and temp string to output
0000 70 .EXTRN STR$COPY_R_R8 ; copy temp str to dest str
0000 71 .EXTRN STR$FREE_DX ; deallocate temp string
0000 72 .EXTRN STR$TRIM ; routine to trim blanks and tabs
0000 73
0000 74 :
0000 75 : MACROS:
0000 76 :
0000 77 :
0000 78 :+
0000 79 : Macro to generate tables to translate ASCII characters. This macro
0000 80 : will generate a table of bytes starting at the designated character
0000 81 : for the series of characters needed.
0000 82 :-
0000 83
0000 84 .MACRO GEN_TRANS_TABLE, NUMBER_OF_CHARS, STARTING_CHAR
0000 85 I = STARTING_CHAR
0000 86 .REPT NUMBER_OF_CHARS
0000 87 .BYTE I
0000 88 I = I + 1
0000 89 .ENDR
0000 90 .ENDM
0000 91
0000 92 :
0000 93 : EQUATED SYMBOLS:
0000 94 :
0000 95 :
00000020 0000 96 blank = ^A/ /
00000001 0000 97 space_or_tab = ^X0001
00000002 0000 98 junk_byte = ^X0002
00000004 0000 99 quote = ^X0004
00000003 0000 100 space_tab_junk = ^X0003
0000 101
00000000 0000 102 trim_parity_bit = 0
00000001 0000 103 disc_sp_tab_bit = 1
00000002 0000 104 disc_junk_bit = 2
00000003 0000 105 disc_lead_bit = 3
00000004 0000 106 compact_bit = 4
00000005 0000 107 upper_case_bit = 5
00000006 0000 108 parens_bit = 6
00000007 0000 109 disc_trail_bit = 7
00000008 0000 110 ignor_quote_bit = 8
```


DECLARATIONS

```
0000 111
0000 112 :
0000 113 : OWN STORAGE:
0000 114 :
0000 115 :
0000 116 :
0000 117 : PSECT DECLARATIONS:
0000 118 :
00000000 119 : .PSECT _BAS$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 120 : EXE, RD, NOWRT, LONG
0000 121
```


BAS\$EDIT - Perform BASIC EDITS function

.SBTTL BAS\$EDIT - Perform BASIC EDITS function

++
FUNCTIONAL DESCRIPTION:

This routine implements the BASIC-PLUS-2 EDITS function. for AST re-entrancy, the routine will create a local dynamic string descriptor and call STR\$COPY to copy the source string to the local. The routine will then use the local string (which will not need to get larger) to do all the editing necessary (since source and destination strings may overlap for the VAX character instructions. After the all the editing is finished, the routine will call STR\$COPY to copy the edited string to the destination string.

CALLING SEQUENCE:

CALL dest_string.wx.dx = BAS\$EDIT (src_string.rx.dx, mask.rwu.v)

INPUT PARAMETERS:

src_string = 8
mask = 12

IMPLICIT INPUTS:

NONE

OUTPUT PARAMETERS:

dest_string = 4

IMPLICIT OUTPUTS:

NONE

FUNCTION VALUE:

COMPLETION CODES:

NONE

SIDE EFFECTS:

This routine calls STR\$COPY and so may have any of its side effects or signal any of its errors. This includes allocating and deallocating dynamic string space, and write locking strings for a short time.

--

.ENTRY BAS\$EDIT, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,IV>

+

Create a local descriptor and copy the input string to it using STR\$COPY

-

51	08	AC	D0	0002	176	MOVL	src_string(AP), R1	; pointer to src string
		7E	D4	0006	177	CLRL	-(SP)	; address of local string
020E0000	8F	DD	0008	178	178	PUSHL	#<<DSC\$K_CLASS_D @ 24> ! <DSC\$K_DTYPE_T @ 16>>	; fill type, class and length
			000E	179				

- Perform BASIC EDITS function

VAX/VMS Macro V04-00
[BASRTL.SRC]BASEDIT.MAR:1Page 5
(3)[illegible]

6-SEP-1984 10:24:23

VAX/VMS Macro V04-00
[BASRTL.SRC]BASEDIT.MAR;1

Page 6
(3)

Address	Hex	Asm	Comment
006F	236	RETURN:	
006F	237	MOVL dest_string(AP), R0	; destination string
0073	238	MOVQ R6, R1	; source str by ref for copy
0076	239	JSB G^STR\$COPY_R_R8	; copy edited to dest
007C	240	RET2: MOVL SP, -(SP)	; address of temp str descr
007F	241	CALLS #1, G^STR\$FREE1_DX	; deallocate temp string
0086	242	CLRQ (SP)+	; pop temp descrptr from stack
0088	243	RET	
0089	244		
0089	245		
0089	246	:+	
0089	247	; Bit 8 is set, so remaining bits do not alter any characters after a	
0089	248	; quote and before a matching quote or the end of the string.	
0089	249	; Scan for quotes and pass unquoted substrings to routine to do editing	
0089	250	; for remaining bits.	
0089	251	;-	
0089	252		
0089	253	INIT_QUOTE_LOOP:	
0089	254	MOVL R7, -(SP)	; save start of string
008C	255	CLRL -(SP)	; space to accumulate edited
008E	256		; string length
008E	257		
008E	258	IGNORE_QUOTED:	
008E	259	SCANC R6, (R7), SCAN_SPAN_TABLE, #quote	; find address of quote
0097	260	MOVQ R0, R9	; save # remaining bytes
009A	261		; save ptr to remaining string
009A	262	SUBL R9, R6	; length of scanned string
009D	263	JSB EDIT\$\$SUBSTRING	; go edit scanned string
00A3	264		; on return, R6 & R7 describe
00A3	265		; edited substring
00A3	266	ADDL2 R6, (SP)	; head of scannedstring still
00A6	267		; abuts processed string, add
00A6	268		; edited scanned
00A6	269		; string len to processed len
00A6	270	TSTL R9	; was end of string found?
00A8	271	BEQLU END_STR_UNQUOTE	; yes, go clean up
00AA	272	MOVCL R9, -(R10), (R6)[R7]	; make remaining string abut
00AF	273		; processed string
00AF	274	ADDL3 R6, R7, R10	; remaining string has been
00B3	275		; moved, point R10 to it
00B3	276	DECL R9	; len of remaining str - quote
00B5	277	LOCC (R10)+, R9, (R10)	; pick up quote character and
00B9	278		; move R10 past it to find the
00B9	279		; matching quote by
00B9	280		; looking in remaining string,
00B9	281		; for len of remaining string
00B9	282	DECL R0	; subtract quote from remainng
00BB	283	BLEQU END_STR_INQUOTE	; quote not found, or quote is
00BD	284		; last character
00BD	285	INCL R1	; point past quote to remainng
00BF	286	MOVQ R0, R6	; reset R6 & R7 to current str
00C2	287	SUBL2 R0, R9	; compute len from 1st quote
00C5	288		; thru 2nd quote
00C5	289	INCL R9	; add 1 for 1st quote
00C7	290	ADDL2 R9, (SP)	; accumulate processed length
00CA	291		; by length between (and
00CA	292		; including) quotes


```

      C2 11 00CA 293      BRB      IGNORE_QUOTED      ; continue processing remainng
      00CC 294
      00CC 295 END_STR_INQUOTE:      ; string ended inside quotes
      59 D6 00CC 296      INCL      R9      ; include 1st quote
      56 8E 59 C1 00CE 297      ADDL3    R9, (SP)+, R6      ; update length of processed
      00D2 298      ; string by remaining length
      00D2 299      ; (quoted length) and reset R6
      57 8E D0 00D2 300      MOVL      (SP)+, R7      ; get processed pointer
      98 11 00D5 301      BRB      RETURN
      00D7 302
      56 8E 7D 00D7 303 END_STR_UNQUOTE:      ; restore R6 & R7 to processed
      00DA 304      MOVQ      (SP)+, R6      ; string
      91 58 07 E1 00DA 305      BBC      #disc_trail_bit, R8, RETURN      ; no more editing, return
      FF72 31 00DE 306      BRW      TRIM_AND_RETURN
      00E1 307
      00E1 308
      00E1 309
      03 58 5B D4 00E1 310 EDIT$$$SUBSTRING:
      01 E1 00E3 311      CLRL      R11      ; mask for SCAN_SPAN loop
      00E7 312      BBC      #disc_sp_tab_bit, R8, DISCARD_JUNK ; don't discard spaces and
      5B 01 D0 00E7 313      ; tabs, try junk
      00EA 314      MOVL      #space_or_tab, R11      ; discard spaces and tabs
      03 58 02 E1 00EA 315 DISCARD_JUNK:
      5B 02 CC 00EE 316      BBC      #disc_junk_bit, R8, SCAN_SPAN_LOOP ; don't discard junk bytes
      00F1 317      XORL2    #junk_byte, R11      ; also discard junk bytes
      5B D5 00F1 318 SCAN_SPAN_LOOP:
      35 13 00F3 319      TSTL      R11
      50 56 7D 00F5 320      BEQLU    COMPACT_SPACE      ; no deletes, go to next bit
      61 50 2A 00F8 321      MOVQ      R6, R0      ; initial values for remaining
      50 D5 0101 322 1$:      SCANC     R0, (R1), SCAN_SPAN_TABLE, R11 ; look for delete character
      25 13 0103 323      TSTL      R0      ; end of string?
      56 50 C2 0105 324      BEQLU    COMPACT_SPACE      ; no deletes, next bit
      54 51 D0 0108 325      SUBL2    R0, R6      ; save length scanned
      5B 50 2B 010B 326      MOVL      R1, R4      ; save ptr to end of scanned
      50 D5 0114 327      SPANC     R0, (R1), SCAN_SPAN_TABLE, R11 ; skip over delete characters
      12 13 0116 328      TSTL      R0      ; end of string?
      56 50 C0 0118 329      BEQLU    COMPACT_SPACE      ; rest are deleted, next bit
      7E 54 D0 011B 330      ADDL2    R0, R6      ; add length not deleted
      7E 50 D0 011E 331      MOVL      R4, -(SP)      ; save remaining pointer
      64 61 50 D0 011E 332      MOVL      R0, -(SP)      ; save remaining length
      50 8E 7D 0121 333      MOVC3    R0, (R1), (R4)      ; overwrite deleted portion
      CE 11 0125 334      MOVQ      (SP)+, R0      ; restore remaining len & ptr
      0128 335      BRB      1$      ; continue deleting
      012A 336
      012A 337 COMPACT_SPACE:
      012A 338      ;\clear bit 4 if bit 1 set\
      3A 58 04 E1 012A 339      BBC      #compact_bit, R8, CHECK_UPCASE
      50 56 7D 012E 340      MOVQ      R6, R0      ; init R0 & R1 for scan
      01 0000C595'EF 61 50 2A 0131 341 1$:      SCANC     R0, (R1), SCAN_SPAN_TABLE, #space_or_tab ; look for space
      50 D5 013A 342      TSTL      R0      ; end of string?
      2A 13 013C 343      BEQLU    CHECK_UPCASE      ; no space, next bit
      56 50 C2 013E 344      SUBL2    R0, R6      ; save length scanned
      54 51 D0 0141 345      MOVL      R1, R4      ; save ptr to end of scanned
      01 00000595'EF 61 50 2B 0144 346      SPANC     R0, (R1), SCAN_SPAN_TABLE, #space_or_tab ; skip spaces & tabs
      84 20 90 014D 347      MOVB     #blank, (R4)+      ; put 1 blank in string
      56 D6 0150 348      INCL      R6      ; len=len+1, added 1 space
      50 D5 0152 349      TSTL      R0      ; end of string?
```


BAS\$EDIT

- Perform BASIC EDIT\$ functio

6-SEP-1984 10:24:23

[BASRTL.SRC]BAS\$EDIT.MAR;1

```

        12 13 0154 350      BEQLU      CHECK_UPCASE      ; rest are compacted, next bit
        56 50 C0 0156 351      ADDL2      R0, R8          ; add length not compacted
        7E 54 D0 0159 352      MOVL       R4, -(SP)       ; save remaining pointer
        7E 50 D0 015C 353      MOVL       R0, -(SP)       ; save remaining length
64      61 50 28 015F 354      MOVCB3     R0, (R1), (R4)   ; overwrite compacted portion
        50 8E 7D 0163 355      MOVQ      (SP)+, R0        ; restore remaining len & ptr
        C9 11 0166 356      BRB          1$              ; continue compacting
        0168 357
        0168 358 CHECK_UPCASE:
        16 58 05 E1 0168 359      BBC      #upper_case_bit, R8, CHECK_PARENS ; do not upcase, next bit
        09 58 06 E1 016C 360      BBC      #parens_bit, R8, ONLY_UPCASE    ; upcase set, parens clear
53      00000495'EF 9E 0170 361      MOVAB     UPCASE_PAREN, R3             ; upcase set, parens set
        14 11 0177 362      BRB          TRANSLATE          ; go translate both
        0179 363
        0179 364 ONLY_UPCASE:
        53      00000295'EF 9E 0179 365      MOVAB     UPCASE_TAB, R3          ; upcase set, parens clear
        OB 11 0180 366      BRB          TRANSLATE          ; go upcase
        0182 367
        0182 368 CHECK_PARENS:
        OE 58 06 E1 0182 369      BBC      #parens_bit, R8, END_EDIT_SUB    ; upcase clear, parens clear
53      00000395'EF 9E 0186 370      MOVAB     SQBRK_TO_PAREN, R3           ; upcase clear, parens set
        018D 371
        018D 372 TRANSLATE:
67      56 63 20 67 56 2E 018D 373      MOVTC     R6, (R7), #blank, (R3), R6, (R7) ; translate string into
        0194 374      ; self using table stored in
        0194 375      ; R3 by above code
        0194 376
        0194 377 END_EDIT_SUB:
        05 0194 378      RSB
        0195 379
        0195 380
                                ; End of BAS$EDIT
```


BASSEDIT

- Perform BASIC EDITS functio

```
0195 382 :+
0195 383 : The following tables are used for the SPANC, SCANC, and MOVTC instructions
0195 384 : used in the BASSEDIT routine
0195 385 :-
0195 386
0195 387
0195 388 :+
0195 389 : Table used to 'remove parity bit' from all characters in the character
0195 390 : string. Used to ensure that the high bit is clear on all characters.
0195 391 :-
0195 392 PARITY_TAB:
0195 393     GEN_TRANS_TABLE 128, 0           ; fill ASCII character set
0215 394     GEN_TRANS_TABLE 128, 0           ; truncate hi bit
0295 395
0295 396 :+
0295 397 : Table used to translate lower case to upper case.
0295 398 :-
0295 399 UPCASE_TAB:
0295 400     GEN_TRANS_TABLE 97, 0           ; fill table to start of lowercase
02F6 401     GEN_TRANS_TABLE 26, 65       ; translate domestic lower to upper
0310 402     GEN_TRANS_TABLE 5, 123        ; translate to self
0315 403     GEN_TRANS_TABLE 96, 128       ; translate to self
0375 404     GEN_TRANS_TABLE 31, 192      ; translate int'l lower to upper
FF 0394 405     .BYTE 255                  ; translate to self
0395 406
0395 407 :+
0395 408 : Table used to translate square brackets to parentheses.
0395 409 :-
0395 410 SQBRAK_TO_PAREN:
0395 411     GEN_TRANS_TABLE 91, 0           ; translate to self
28 03F0 412     .BYTE 40                  ; translate [ to (
5C 03F1 413     .BYTE 92                  ; translate to self
29 03F2 414     .BYTE 41                  ; translate ] to )
03F3 415     GEN_TRANS_TABLE 162, 94      ; translate remainder to self
0495 416
0495 417 :+
0495 418 : Table used to translate lower case to upper case AND square brackets
0495 419 : to parentheses.
0495 420 :-
0495 421 UPCASE_PAREN:
0495 422     GEN_TRANS_TABLE 91, 0           ; translate to self
28 04F0 423     .BYTE 40                  ; translate [ to (
5C 04F1 424     .BYTE 92                  ; translate to self
29 04F2 425     .BYTE 41                  ; translate ] to )
04F3 426     GEN_TRANS_TABLE 3, 94        ; translate to self
04F6 427     GEN_TRANS_TABLE 26, 65       ; translate domestic lower to upper
0510 428     GEN_TRANS_TABLE 5, 123        ; translate to self
0515 429     GEN_TRANS_TABLE 96, 128       ; translate to self
0575 430     GEN_TRANS_TABLE 31, 192      ; translate int'l lower to upper
FF 0594 431     .BYTE 255                  ; translate to self
0595 432
0595 433 :+
0595 434 : Table used to discard <CR>, <LF>, <FF>, <DEL>, <NUL>.
0595 435 :-
0595 436 SCAN_SPAN_TABLE:
02 0595 437     .BYTE junk_byte               ; NUL
0596 438     .BYTE 0[8]                   ; skip over
```


BASSEDIT
1-009

E 2

15-SEP-1984 23:38:01
6-SEP-1984 10:24:23

VAX/VMS Macro V04-00
[BASRTL.SRC]BASEDIT.MAR;1

Page 10
(4)

BASSEDIT

- Perform BASIC EDITS functio

```
01 059E 439 .BYTE space_or_tab ; tab
02 059F 440 .BYTE junk_byte ; LF
00 05A0 441 .BYTE 0 ; skip over
02 05A1 442 .BYTE junk_byte[2] ; FF and CR
00 05A3 443 .BYTE 0[13] ; skip over
00 05AF
02 05B0 444 .BYTE junk_byte ; ESC
00 05B1 445 .BYTE 0[4] ; skip over
01 05B5 446 .BYTE space_or_tab ; space
00 05B6 447 .BYTE 0 ; skip over
04 05B7 448 .BYTE quote ; double quote
00 05B8 449 .BYTE 0[4] ; skip over
04 05BC 450 .BYTE quote ; single quote
00 05BD 451 .BYTE 0[87] ; skip over
00 05C9
00 05D5
00 05E1
00 05ED
00 05F9
00 0605
00 0611
02 0614 452 .BYTE junk_byte ; DEL
00 0615 453 .BYTE 0[128] ; skip over
00 0621
00 062D
00 0639
00 0645
00 0651
00 065D
00 0669
00 0675
00 0681
00 068D
00 0695 454 .END
```


BASEDIT
Symbol table

BASEDIT	00000000	RG	02
BLANK	= 00000020		
CHECK_PARENS	00000182	R	02
CHECK_QUOTE_BIT	00000041	R	02
CHECK_UPCASE	00000168	R	02
COMPACT_BIT	= 00000004		
COMPACT_SPACE	0000012A	R	02
DEST_STRING	= 00000004		
DISCARD_JUNK	000000EA	R	02
DISCARD_LEADING	00000031	R	02
DISC_JUNK_BIT	= 00000002		
DISC_LEAD_BIT	= 00000003		
DISC_SP_TAB_BIT	= 00000001		
DISC_TRAIL_BIT	= 00000007		
DSCSA_POINTER	= 00000004		
DSCSK_CLASS_D	= 00000002		
DSCSK_CLASS_S	= 00000001		
DSCSK_DTYPE_T	= 0000000E		
DSCSW_LENGTH	= 00000000		
EDIT\$\$SUBSTRING	000000E1	R	02
END_EDIT_SUB	00000194	R	02
END_STR_INQUOTE	000000CC	R	02
END_STR_UNQUOTE	000000D7	R	02
IGNORE_QUOTED	0000008E	R	02
IGNOR_QUOTE_BIT	= 00000008		
INIT_QUOTE_COOP	00000089	R	02
JUNK_BYTE	= 00000002		
MASK	= 0000000C		
ONLY_UPCASE	00000179	R	02
PARENS_BIT	= 00000006		
PARITY_TAB	00000195	R	02
QUOTE	= 00000004		
RET2	0000007C	R	02
RETURN	0000006F	R	02
SCAN_SPAN_LOOP	000000F1	R	02
SCAN_SPAN_TABLE	00000595	R	02
SPACE_OR_TAB	= 00000001		
SQBRAK_TO_PAREN	00000395	R	02
SRC_STRING	= 00000008		
STR\$COPY_DX_R8	*****	X	00
STR\$COPY_R_R8	*****	X	00
STR\$FREED_DX	*****	X	00
STR\$TRIM	*****	X	00
TRANSLATE	0000018D	R	02
TRIM_AND_RETURN	00000053	R	02
TRIM_PARITY_BIT	= 00000000		
UPCASE_PAREN	00000495	R	02
UPCASE_TAB	00000295	R	02
UPPER_CASE_BIT	= 00000005		

-----+
! Psect synopsis !
-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_BAS\$CODE	00000695 (1685.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

-----+
! Performance indicators !
-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:00.61
Command processing	125	00:00:00.46	00:00:03.81
Pass 1	238	00:00:08.67	00:00:17.27
Symbol table sort	0	00:00:00.22	00:00:00.62
Pass 2	93	00:00:02.08	00:00:04.02
Symbol table output	7	00:00:00.09	00:00:00.47
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	497	00:00:11.62	00:00:26.82

The working set limit was 1200 pages.
58789 bytes (115 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 177 non-local and 2 local symbols.
454 source lines were read in Pass 1, producing 16 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.

-----+
! Macro library statistics !
-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4

190 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:BASEDIT/OBJ=OBJ\$:BASEDIT MSRC\$:BASEDIT/UPDATE=(ENH\$:BASEDIT)

0022 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASEDDFS
LIS

BASERROR
LIS

BASEDDDF
LIS

BASEDIT
LIS

BASEND
LIS

BASEDUP
LIS

BASEMULP
LIS

BASEDDGB
LIS

BASERTXT
LIS